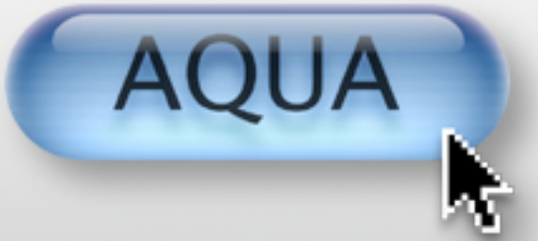
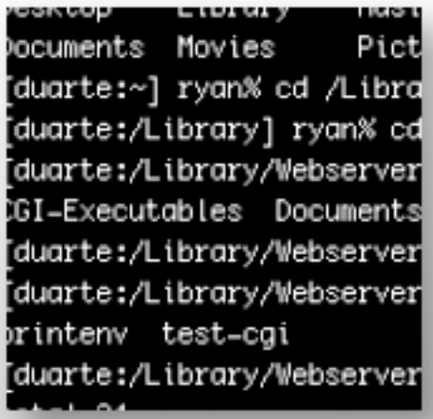
Optimizing Carbon Performance

Session 416





Optimizing Carbon Performance

Xavier Legros
Mac OS X Evangelist
xavier@apple.com

Carbon Performance

- Performance is a combination of
  - Real Performance
  - Perceived Performance

Optimizing Carbon Performance

Curt Rothert

What You'll Learn

- Learn techniques for identifying problem areas
- Learn techniques for improving the speed and responsiveness of your application
- Learn how to use modern technologies instead of relying on antiquated techniques

Performance Problem Areas

- Slow application launch
- Low level services
- Slow drawing
- Poor responsiveness

Application Launch Performance

What's Going on During Launch?

- File I/O
  - Binary
  - Resources
  - Preferences
  - Plug-ins
- Application initialization
  - Allocations
  - Table initialization, etc.

How To Find Launch Problems

- Use Sampler during launch
  - Find where all your CPU time is being spent
- Use fs_usage to determine file system activity
- Inspect your initialization code

Launch Solutions

- Compile: optimize for size
- Provide feedback to the user
  - Put up a splash screen
  - Put up main window as early as possible
- Avoid I/O during launch
- “Lazy” initialization
  - Defer allocation
  - Avoid static class construction

Lazy Initialization

- Rather than initialize a component in anticipation of use, only initialize when the component will be used

```
static bool     sMyComponentInitialized = false;
....

if ( sMyComponentInitialized == false )
{
        InitializeMyComponent();
        sMyComponentInitialized = true;
}
```

Threading, Memory and I/O

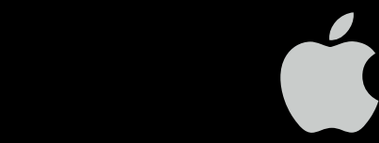
John Iarocci

Threading—APIs

- Cooperative Threads (Carbon Thread Manager)
  - Requires YieldToAnyThread() to work well
  - Use Carbon Event Timers or MP threads or pthreads
- MPThreads and pthreads
  - Use them when it makes sense
  - Think about the amount of work done on the thread
  - Consider the cost of IPC and locking

Threading—Tips

- Use gdb
  - `thread apply all, info thread(s)`
  - `mach_thread_self() == local thread`
- Get familiar with top
  - How many threads are you using?
  - Use “top -c” to show context switches
- Use /Developer/Applications/ThreadViewer
  - Visualize threads
  - Examine memory usage, lock usage, and priority



Memory—APIs

- Think about your use of Handles
  - HLock and HUnlock are not needed
  - SetHandleSize does change a Handle
- Use malloc & free where you can
  - Memory Manager is layered on top of malloc & free

Memory—Tips

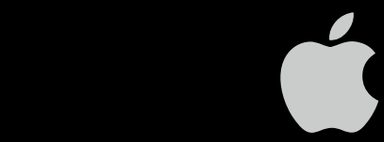
- Use Sampler and sample
  - Look for HLock, HUnlock, HSetState, HGetState
- Use MallocDebug and leaks
  - Leaks
- Use “top -w” to show deltas

Files—APIs

- Use `FSRef` based API
- Only ask for what you need
  - `FSGetCatalogInfo`
  - `FSGetVolumeInfo`
- Test your app on POSIX volume
  - UFS, SMB, NFS, WebDAV
  - Test your app with a network home directory

Files—Tips

- Use `fs_usage` to verify your app's performance
  - `fs_usage -f filesys` to just look at filesystem
  - Use `DYLD_IMAGE_SUFFIX = _debug` to enable FileManager tracing
- Use large page-aligned buffers for optimal I/Os
- Use noCacheBit to avoid purging system cache



Networking—APIs

- Use CFNetwork
  - Use asynchronous API
  - Use large buffers
    - Match the socket buffer size
    - Adapt buffer size based on throughput
- Don’t use:
  - OpenTransport
    - Use either sockets or CFNetwork
  - URLAccess
    - Use either CFNetwork or NSURLRequest/Response

Networking—Tips

- Measure your use of the network
  - Use fs_usage -f network
  - Use tcpdump to examine packets on local network

Drawing

Drawing Performance Problems

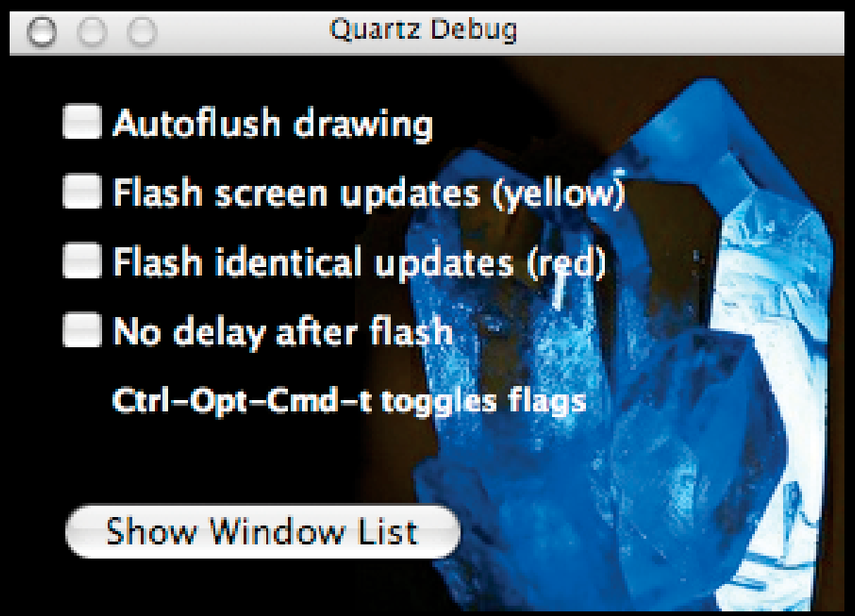
- Too much drawing
- Inefficient drawing
- Text performance

Too Much Drawing

- Scope
  - Drawing more than needs to be drawn
- Frequency
  - Drawing periodically based on some state that may not change
  - Drawing based on event stream that doesn’t properly reflect what you are interested in

Am I Drawing Too Much?

- Use Quartz Debug
  - '/Developer/Applications/Quartz Debug'
- Works with all Carbon drawing



Drawing Solutions:

- Only draw what you need to draw
- Only draw when you need to draw
- Adopt HIView

HIView

- Aka Compositing Mode
- Designed to draw as little as possible
  - Invalidation model
- Can specify opacity
- See session 425—Carbon: HIView in Depth

Draw Smarter If Not Composited

- Hide controls around multi state changes
  - Set min/max/value
  - Adding a number of items to a Popup control
- Don’t redraw things that are valid
  - Pay attention to update region in the update event

Inefficient Drawing

Am I Drawing Efficiently?

- Sample
- Inspect your drawing code
- Here’s some stuff to look for…

Triple Buffering

- Mac OS X already double buffers
- Eliminate application double buffering

Region Processing

- Use rectangles rather than complex regions
- No 64k limit any more
- Better performance
- If dirty region maintenance shows up in samples:
  - Simplify dirty region (but not too much)
  - Use QDSetDirtyRegion

Redundant Calculations

- Cache frequently calculated results
  - Text dimensions
  - Part sizes
  - Example: Tab control

Mixing QuickDraw and Quartz

- QD port must be synchronized with Quartz Context
- Avoid mixing Quartz and QuickDraw drawing
  - If you must mix, draw in blocks for each framework:

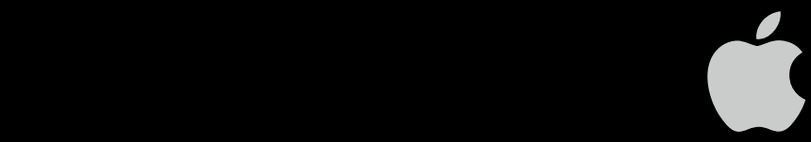
```
void Draw()
{
        DrawWithQDAPIs()
        DrawWithQuartzAPIs()
}
```

Use Quartz

- Do as much drawing in between Begin/End context
- Convert QD formats to native CG formats
  - Example: convert PICT to PDF using QDPictDrawToCGContext
  - Either at build time or at run time and reuse this format

Getting a CGContext

- QDBeginCGContext/QDEndCGContext
  - Caches CGContext—you don't have to
  - Allows to print both CG and QD on the same page
  - (But QD disabled between Begin/End)
- CGContext and GrafPort are not synchronized
  - SyncCGContextOriginWithPort



Flush vs. Synchronize

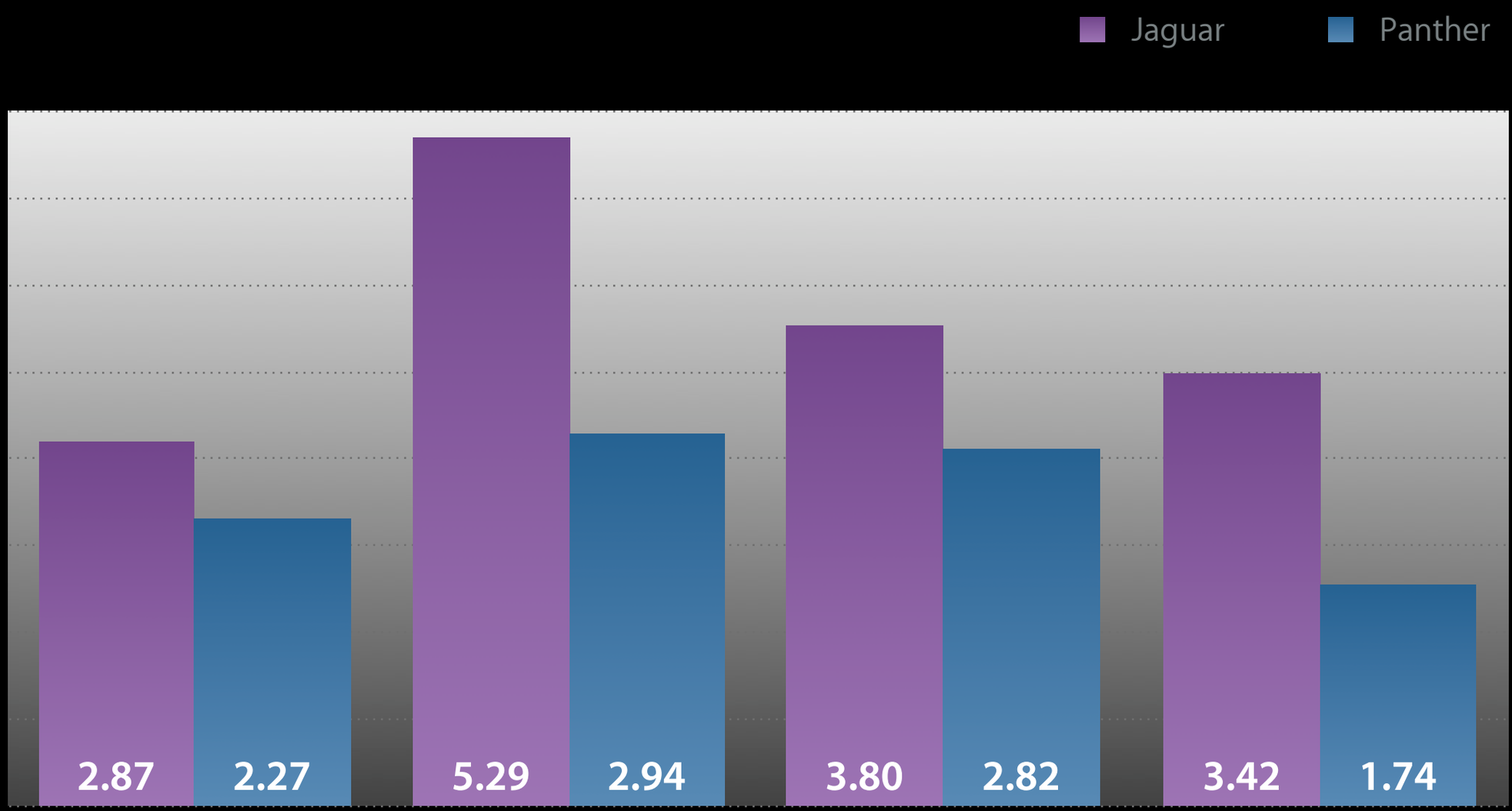
- Avoid over-flushing the context
- Use CGContextFlush for immediate response
- Use CGContextSynchronize for delayed flush
  - And let HIToolbox do the flushing in the RunLoop

Use HITheme APIs

- Simple mapping from Appearance to HITheme APIs
- Check out HITheme.h
- All require a CGContextRef
- All require an orientation
  - Normal (top-left origin)
  - Inverted (bottom-left origin)
- Check out Session 409—Carbon: The HIToolbox

HITheme APIs and Performance

- Using these APIs for drawing improve performance by 45%
- Most common controls average 25% faster

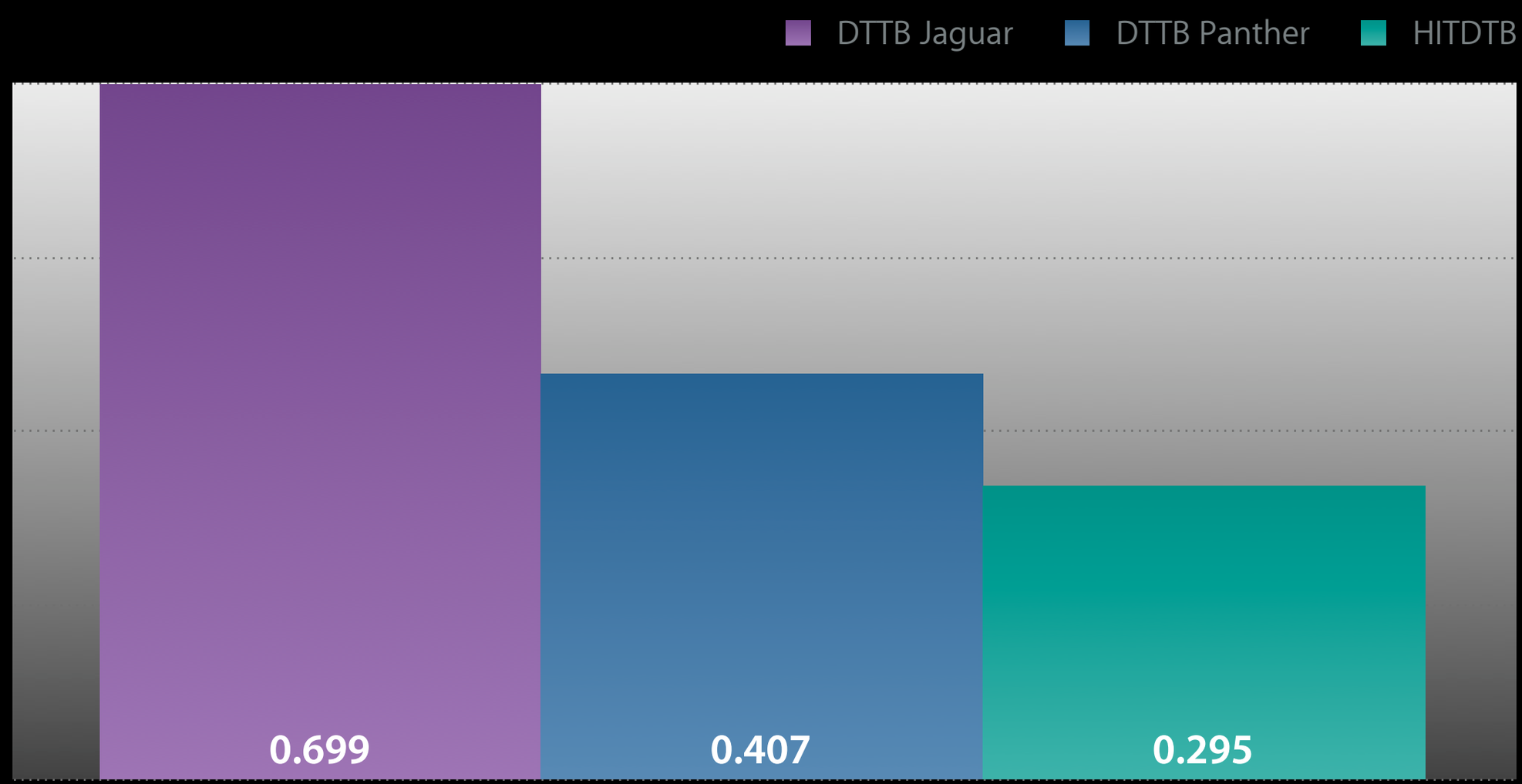


Text Drawing

SPEEDY

User Interface Text

- HIThemeGetTextDimensions
- HIThemeDrawTextBox



ATSUI Performance Tips

- Create a layout for each paragraph when providing editable text features
- Reuse layout and style objects where possible
- Use ATSUBatchBreakLine where possible

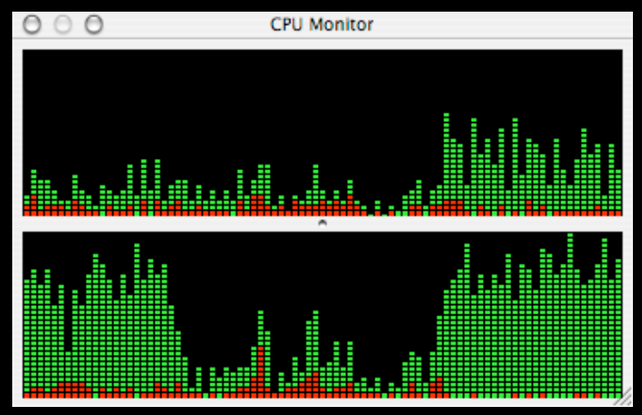
ATSUI Performance Tips (Cont.)

- Use ATSUGetGlyphBounds to get the layout bounds
- Create and use ATSUFontFallback objects if needed instead of relying on the global fallbacks
- Cache and share ATSUStyles among ATSUTextLayout objects

Responsiveness

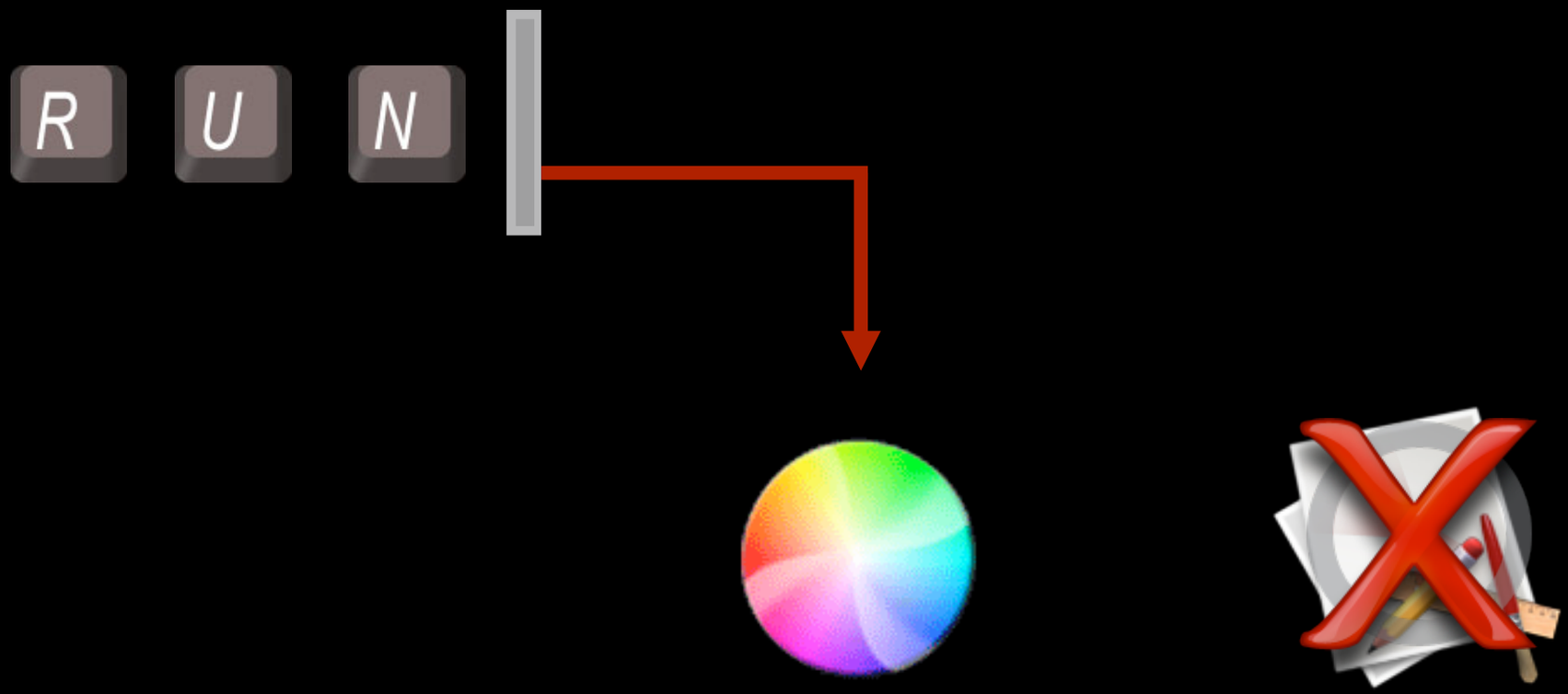
Responsiveness Issues

- Spinning cursor
- Parts of application non-responsive while tracking mouse
- CPU pegged performing “simple” tasks
- Application is slow to respond to user interaction
- How to find problems:
  - /Applications/Utilities/Activity Monitor
  - /Developer/Applications/Spin Controls
  - top



Why Do I Get the Spinning Cursor?

- Events are not being removed from the event queue

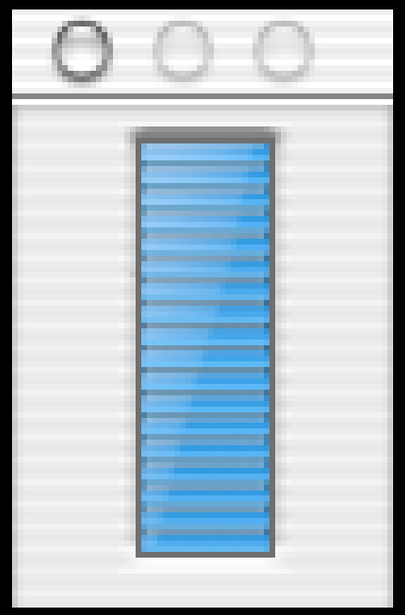


Solutions for Poor Response

- Call the Event Loop
- Adopt Carbon Events
- Thread your app
- Watch for polling…

Am I Polling?

- Watch if the CPU is pegged when doing benign actions
- /Applications/Utilities/Activity Monitor
- top



Alternatives to Polling:

- You need to track the cursor
  - While (MouseDown()) GetMouse();
- Need to determine where the cursor is
  - Poll cursor location
- To react to a condition that may have changed
  - Check a seed value
- To “idle” for a specified amount of time
  - Spin until timeout

Alternatives to Polling:

- You need to track the cursor
  - Use TrackMouseLocation
- Need to determine where the cursor is
  - Poll cursor location
- To react to a condition that may have changed
  - Check a seed value
- To “idle” for a specified amount of time
  - Spin until timeout

Alternatives to Polling:

- You need to track the cursor

  - Use TrackMouseLocation
- Need to determine where the cursor is

  - Use Tracking Regions
- To react to a condition that may have changed
  - Check a seed value
- To “idle” for a specified amount of time
  - Spin until timeout

Alternatives to Polling:

- You need to track the cursor
  - Use TrackMouseLocation
- Need to determine where the cursor is
  - Use Tracking Regions
- To react to a condition that may have changed
  - Watch for Notifications
- To “idle” for a specified amount of time
  - Spin until timeout

Alternatives to Polling:

- You need to track the cursor
  - Use TrackMouseLocation
- Need to determine where the cursor is
  - Use Tracking Regions
- To react to a condition that may have changed
  - Watch for Notifications
- To “idle” for a specified amount of time
  - Use timers or jump into the run loop

Asynchronous Window Dragging

- New facility to allow the window server to handle dragging
- Frees the app from dealing with this
- Even when the app is busy, the user can move windows around

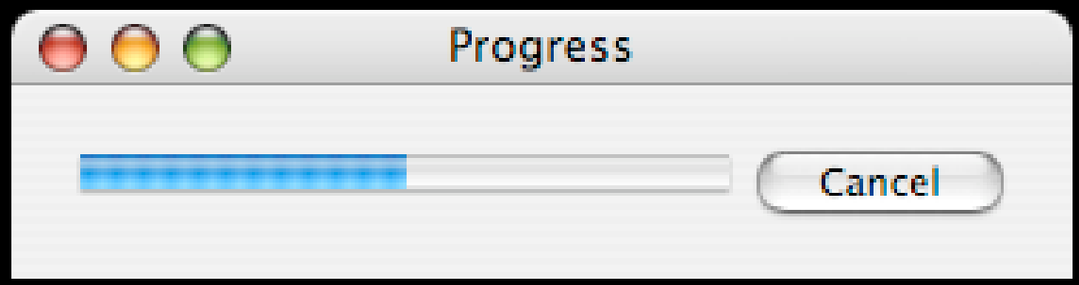
Demo

Guy Fullerton

Male Exp3; Cr 2; Medium-size Humanoid (Human)

General Tips

- Cache
  - Be aware of time/space tradeoff
- Always provide feedback
  - Progress Bars
  - Chasing Arrows
  - Allow cancellation



Summary

Use Performance Tools

- Quartz Debug
- Sampler
- Shark
- Spin Control

Use Modern APIs

- Quartz
- CFNetwork
- Carbon Events
- HIView
- HITheme

Performance is Everybody's Job!

Roadmap

409 Carbon: The HIToolbox Architectural overview of the HIToolbox framework	**Presidio** **Wed, 9:00am**
423 How to Write a Modern Carbon App Learn to create a native Mac OS X app using C/C++	**Nob Hill** **Fri, 2:00pm**
425 Carbon: HIView in Depth Learn architecture of Carbon view system	**Presidio** **Fri, 3:30pm**
305 Tuning Software with Performance Tools	**Presidio** **Wed, 5:00pm**
310 Debugging and Tuning Carbon Applications	**Presidio** **Fri, 9:00am**

Who to Contact

Xavier Legros
Fabulous Favorite French Guy
xavier@apple.com

http://developer.apple.com/wwdc2003/urls.html

Reference Library

Documentation

Performance Documentation

Reference Library

Technical Notes

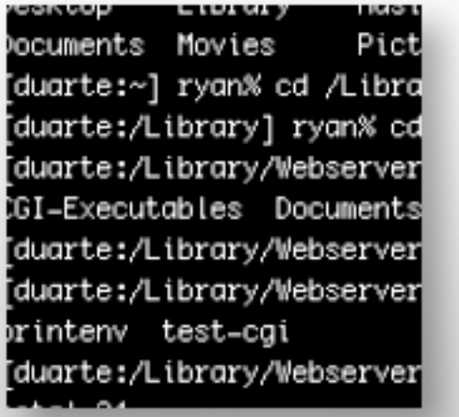
TN2051: Mac OS X QuickDraw Performance
http://developer.apple.com/technotes/tn/tn2051.html

FL16: File Manager Performance and Caching
http://developer.apple.com/technotes/fl/fl_16.html

Technical Q&As

QA1027: Improving ATSUI Text Drawing
http://developer.apple.com/qa/qa2001/qa1027.html

Q&A



Xavier Legros

Mac OS X Evangelist, Favorite French Guy

xavier@apple.com

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference